HT16K33 Driver for Apache Mynewt

Generated by Doxygen 1.8.14

Contents

Chapter 1

HT16K33 14-Segment Display Driver for Apache Mynewt

The main package documentation for this repository is available at microbuilder.io/mb_ht16k33.

It has been written against Apache Mynewt 1.4.1.

Folder Structure

- Reasonably complete **documentation** is available in the docs folder.

Hardware

This package has been tested with the following hardware:

- Adafruit Feather nRF52 Pro with myNewt Bootloader - nRF52832
- Adafruit 14-Segment Alphanumeric LED FeatherWing
- Dual Alphanumeric Display -Yellow 0.54" Digit Height - Pack of 2

Usage

The following code snippet shows how this package can be used. It assumes the presence of 4 14-segment displays, numbered 0..3 below:

```
int rc;

/* Initialise the display at brightness 9/16. */
rc = ht16k33_init(HTK16K33_REG_DIM_9);
if (rc) {
    console_printf("HT16K33 init failed: %d\n", rc);
}

/* Display some test data on the display. */
ht16k33_write_num(0, 1, true);      /* 1. */
ht16k33_write_num(1, 2, false);     /* 2  */
ht16k33_write_num(2, 5, false);     /* 5  */
ht16k33_write_alpha(3, 'K', false); /* K  */
```

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Module Documentation

4.1 ht15k33

Modules

- Defines
- Functions

4.1.1 Detailed Description

Functions to work with the HT16K33 14-segment display driver.

4.2 Defines

Macros

- #define HT16K33_REG_CMD_MODE_ON (0b00100001)
- #define HT16K33_REG_CMD_MODE_STDBY (0b00100000)
- #define HT16K33_REG_DISP_ON_BLINK_NONE (0b10000001)
- #define HT16K33_REG_DISP_ON_BLINK_2HZ (0b10000011)
- #define HT16K33_REG_DISP_ON_BLINK_1HZ (0b10000101)
- #define HT16K33_REG_DISP_ON_BLINK_0_5HZ (0b10000111)
- #define HT16K33_REG_DISP_OFF (0b00000000)
- #define HTK16K33_REG_DIM_1 (0b11100000)
- #define HTK16K33_REG_DIM_2 (0b11100001)
- #define HTK16K33_REG_DIM_3 (0b11100010)
- #define HTK16K33_REG_DIM_4 (0b11100011)
- #define HTK16K33_REG_DIM_5 (0b11100100)
- #define HTK16K33_REG_DIM_6 (0b11100101)
- #define HTK16K33_REG_DIM_7 (0b11100110)
- #define HTK16K33_REG_DIM_8 (0b11100111)
- #define HTK16K33_REG_DIM_9 (0b11101000)
- #define HTK16K33_REG_DIM_10 (0b11101001)
- #define HTK16K33_REG_DIM_11 (0b11101010)
- #define HTK16K33_REG_DIM_12 (0b11101011)
- #define HTK16K33_REG_DIM_13 (0b11101100)
- #define HTK16K33_REG_DIM_14 (0b11101101)
- #define HTK16K33_REG_DIM_15 (0b11101110)
- #define HTK16K33_REG_DIM_16 (0b11101111)

4.2.1 Detailed Description

HTK16K33-related public DEFINEs.

4.2.2 Macro Definition Documentation

4.2.2.1 HT16K33_REG_CMD_MODE_ON

```
#define HT16K33_REG_CMD_MODE_ON (0b00100001)
```

Sets the IC operating mode to ON via the COMMAND register.

Definition at line 41 of file ht16k33.h.

4.2.2.2 HT16K33_REG_CMD_MODE_STDBY

```
#define HT16K33_REG_CMD_MODE_STDBY (0b00100000)
```

Sets the IC operating mode to STANDBY via the COMMAND register.

Definition at line 44 of file ht16k33.h.

4.2.2.3 HT16K33_REG_DISP_OFF

```
#define HT16K33_REG_DISP_OFF (0b00000000)
```

Turns the display off.

Definition at line 59 of file ht16k33.h.

4.2.2.4 HT16K33_REG_DISP_ON_BLINK_0_5HZ

```
#define HT16K33_REG_DISP_ON_BLINK_0_5HZ (0b10000111)
```

Turns the display on and sets blinky rate to 0.5Hz.

Definition at line 56 of file ht16k33.h.

4.2.2.5 HT16K33_REG_DISP_ON_BLINK_1HZ

```
#define HT16K33_REG_DISP_ON_BLINK_1HZ (0b10000101)
```

Turns the display on and sets blinky rate to 1Hz.

Definition at line 53 of file ht16k33.h.

4.2.2.6 HT16K33_REG_DISP_ON_BLINK_2HZ

```
#define HT16K33_REG_DISP_ON_BLINK_2HZ (0b10000011)
```

Turns the display on and sets blinky rate to 2Hz.

Definition at line 50 of file ht16k33.h.

4.2.2.7 HT16K33_REG_DISP_ON_BLINK_NONE

```
#define HT16K33_REG_DISP_ON_BLINK_NONE (0b10000001)
```

Turns the display on and sets blinky rate to 0.

Definition at line 47 of file ht16k33.h.

4.2.2.8 HTK16K33_REG_DIM_1

```
#define HTK16K33_REG_DIM_1 (0b11100000)
```

Definition at line 62 of file ht16k33.h.

4.2.2.9 HTK16K33_REG_DIM_10

```
#define HTK16K33_REG_DIM_10 (0b11101001)
```

Definition at line 89 of file ht16k33.h.

4.2.2.10 HTK16K33_REG_DIM_11

```
#define HTK16K33_REG_DIM_11 (0b11101010)
```

Definition at line 92 of file ht16k33.h.

4.2.2.11 HTK16K33_REG_DIM_12

```
#define HTK16K33_REG_DIM_12 (0b11101011)
```

Definition at line 95 of file ht16k33.h.

4.2.2.12 HTK16K33_REG_DIM_13

```
#define HTK16K33_REG_DIM_13 (0b11101100)
```

Definition at line 98 of file ht16k33.h.

4.2.2.13 HTK16K33_REG_DIM_14

```
#define HTK16K33_REG_DIM_14 (0b11101101)
```

Definition at line 101 of file ht16k33.h.

4.2.2.14 HTK16K33_REG_DIM_15

```
#define HTK16K33_REG_DIM_15 (0b11101110)
```

Definition at line 104 of file ht16k33.h.

4.2.2.15 HTK16K33_REG_DIM_16

```
#define HTK16K33_REG_DIM_16 (0b11101111)
```

Definition at line 107 of file ht16k33.h.

4.2.2.16 HTK16K33_REG_DIM_2

```
#define HTK16K33_REG_DIM_2 (0b11100001)
```

Definition at line 65 of file ht16k33.h.

4.2.2.17 HTK16K33_REG_DIM_3

```
#define HTK16K33_REG_DIM_3 (0b11100010)
```

Definition at line 68 of file ht16k33.h.

4.2.2.18 HTK16K33_REG_DIM_4

```
#define HTK16K33_REG_DIM_4 (0b11100011)
```

Definition at line 71 of file ht16k33.h.

4.2.2.19 HTK16K33_REG_DIM_5

```
#define HTK16K33_REG_DIM_5 (0b11100100)
```

Definition at line 74 of file ht16k33.h.

4.2.2.20 HTK16K33_REG_DIM_6

```
#define HTK16K33_REG_DIM_6 (0b11100101)
```

Definition at line 77 of file ht16k33.h.

4.2.2.21 HTK16K33_REG_DIM_7

```
#define HTK16K33_REG_DIM_7 (0b11100110)
```

Definition at line 80 of file ht16k33.h.

4.2.2.22 HTK16K33_REG_DIM_8

```
#define HTK16K33_REG_DIM_8 (0b11100111)
```

Definition at line 83 of file ht16k33.h.

4.2.2.23 HTK16K33_REG_DIM_9

```
#define HTK16K33_REG_DIM_9 (0b11101000)
```

Definition at line 86 of file ht16k33.h.

4.3 Functions

Functions

- int ht16k33_init (uint8_t brightness)
- int ht16k33_clear (void)
- int ht16k33_write_cmd (uint8_t reg)
- int ht16k33_write_num (uint8_t addr, uint8_t value, bool dec)
- int ht16k33_write_hex (uint8_t addr, uint8_t value, bool dec)
- int ht16k33_write_alpha (uint8_t addr, uint8_t value, bool dec)

4.3.1 Detailed Description

Functions to work with the HT16K33.

4.3.2 Function Documentation

4.3.2.1 ht16k33_clear()

```
int ht16k33_clear (
            void  )
```

Clears the display.

Returns

0 on success, an error code on error.

Definition at line 292 of file ht16k33.c.

```
00293 {
00294     int rc;
00295
00296     /* Clear the buffer. */
00297     memset(g_ht16k33_buffer_16, 0, sizeof(g_ht16k33_buffer_16));
00298
00299     rc = ht16k33_i2c_writelen(g_ht16k33_buffer_16,
00300         sizeof(g_ht16k33_buffer_16));
00301     if (rc != 0) {
00302         goto err;
00303     }
00304
00305     return 0;
00306 err:
00307     return rc;
00308 }
```

4.3.2.2 ht16k33_init()

```
int ht16k33_init (
            uint8_t brightness )
```

Initialises and enables the display controller and the display.

Parameters

brightness	The brightness register value to apply.

Returns

0 on success, an error code on error.

Definition at line 240 of file ht16k33.c.

```
00241 {
00242     int rc;
00243
00244     /* Insert a short delay to allow the IC to start up. */
00245     os_time_delay((1 * OS_TICKS_PER_SEC)/1000);
00246
00247     /* Initialise the stats entry. */
00248     rc = stats_init(
00249         STATS_HDR(g_ht16k33stats),
00250         STATS_SIZE_INIT_PARMS(g_ht16k33stats, STATS_SIZE_32),
00251         STATS_NAME_INIT_PARMS(ht16k33_stat_section));
00252     if (rc != 0) {
00253         goto err;
00254     }
00255
00256     /* Register the entry with the stats registry. */
00257     rc = stats_register("ht16k33", STATS_HDR(g_ht16k33stats));
00258     if (rc != 0) {
00259         goto err;
00260     }
00261
00262     /* Enable the controller. */
00263     rc = ht16k33_write_cmd(HT16K33_REG_CMD_MODE_ON);
00264     if (rc != 0) {
00265         goto err;
00266     }
00267
00268     /* Clear the display. */
00269     rc = ht16k33_clear();
00270     if (rc != 0) {
00271         goto err;
00272     }
00273
00274     /* Turn the display on. */
00275     rc = ht16k33_write_cmd(HT16K33_REG_DISP_ON_BLINK_NONE);
00276     if (rc != 0) {
00277         goto err;
00278     }
00279
00280     /* Set the display brightness */
00281     rc = ht16k33_write_cmd(brightness);
00282     if (rc != 0) {
00283         goto err;
00284     }
00285
00286     return (0);
00287 err:
00288     return (rc);
00289 }
```

4.3.2.3 ht16k33_write_alpha()

```
int ht16k33_write_alpha (
            uint8_t addr,
            uint8_t value,
            bool dec )
```

Writes an alpha-numeric value to the specified address on the HT16K33.

Parameters

addr	The address to write data to.
value	The alpha-numeric value to write to 'addr'.
dec	Whether or not to display the decimal point.

Returns

0 on success, an error code on error.

Definition at line 379 of file ht16k33.c.

```
00380 {
00381     int rc;
00382
00383     /* Make sure we stay in range. */
00384     if (value >= 128) {
00385         rc = OS_EINVAL;
00386         goto err;
00387     }
00388
00389     /* Set the address and hex data. */
00390     g_ht16k33_buffer_16[addr*2+1] = (g_ht16k33_tbl_alpha[value] & 0xFF);
00391     g_ht16k33_buffer_16[addr*2+2] = (g_ht16k33_tbl_alpha[value] >> 8) & 0xFF;
00392
00393     /* Add the decimal point to the output if requested. */
00394     if (dec) {
00395         g_ht16k33_buffer_16[addr*2+2] |= 0x40;
00396     }
00397
00398     rc = ht16k33_i2c_writelen(g_ht16k33_buffer_16, sizeof(g_ht16k33_buffer_16));
00399     if (rc != 0) {
00400         goto err;
00401     }
00402
00403     return 0;
00404 err:
00405     return rc;
00406 }
```

4.3.2.4 ht16k33_write_cmd()

```
int ht16k33_write_cmd (
              uint8_t reg )
```

Writes a command byte to the HT16K33.

Parameters

reg	The register value to write.

Returns

0 on success, an error code on error.

Definition at line 311 of file ht16k33.c.

```
00312 {
00313     int rc;
00314
00315     rc = ht16k33_i2c_write8(reg);
00316     if (rc != 0) {
00317         goto err;
00318     }
00319
00320     return 0;
00321 err:
00322     return rc;
00323 }
```

4.3.2.5 ht16k33_write_hex()

```
int ht16k33_write_hex (
             uint8_t addr,
             uint8_t value,
             bool dec )
```

Writes a hexadecimal value to the specified address on the HT16K33.

Parameters

addr	The address to write data to.
value	The hexadecimal value to write to 'addr'.
dec	Whether or not to display the decimal point.

Returns

0 on success, an error code on error.

Definition at line 356 of file ht16k33.c.

```
00357 {
00358     int rc;
00359
00360     /* Make sure we stay in range. */
00361     if (value > 0xF) {
00362         rc = OS_EINVAL;
00363         goto err;
00364     }
00365
00366     if (value < 10) {
00367         /* Handle decimal values 0..9 */
00368         return ht16k33_write_num(addr, value, dec);
00369     } else {
00370         /* Display HEX value as ASCII */
00371         return ht16k33_write_alpha(addr, value+55, dec);
00372     }
00373
00374 err:
00375     return rc;
00376 }
```

4.3.2.6 ht16k33_write_num()

```
int ht16k33_write_num (
            uint8_t addr,
            uint8_t value,
            bool dec )
```

Writes a decimal value to the specified address on the HT16K33.

Parameters

addr	The address to write data to.
value	The decimal value to write to 'addr'.
dec	Whether or not to display the decimal point.

Returns

0 on success, an error code on error.

Definition at line 326 of file ht16k33.c.

```
00327 {
00328     int rc;
00329
00330     /* Make sure we stay in range. */
00331     if (value > 9) {
00332         rc = OS_EINVAL;
00333         goto err;
00334     }
00335
00336     /* Set the address and hex data. */
00337     g_ht16k33_buffer_16[addr*2+1] = (g_ht16k33_tbl_alpha[value + 48] & 0xFF);
00338     g_ht16k33_buffer_16[addr*2+2] = (g_ht16k33_tbl_alpha[value + 48]>>8) & 0xFF;
00339
00340     /* Add the decimal point to the output if requested. */
00341     if (dec) {
00342         g_ht16k33_buffer_16[addr*2+2] |= 0x40;
00343     }
00344
00345     rc = ht16k33_i2c_writelen(g_ht16k33_buffer_16, sizeof(g_ht16k33_buffer_16));
00346     if (rc != 0) {
00347         goto err;
00348     }
00349
00350     return 0;
00351 err:
00352     return rc;
00353 }
```

Chapter 5

File Documentation

5.1 /Users/kevin/Dropbox/microBuilder/Code/nRF52/Mynewt/garden/repos/mb_ht16k33/include/ht16k33 File Reference

```
#include <stdint.h>
#include "os/mynewt.h"
```

Macros

- #define HT16K33_REG_CMD_MODE_ON (0b00100001)
- #define HT16K33_REG_CMD_MODE_STDBY (0b00100000)
- #define HT16K33_REG_DISP_ON_BLINK_NONE (0b10000001)
- #define HT16K33_REG_DISP_ON_BLINK_2HZ (0b10000011)
- #define HT16K33_REG_DISP_ON_BLINK_1HZ (0b10000101)
- #define HT16K33_REG_DISP_ON_BLINK_0_5HZ (0b10000111)
- #define HT16K33_REG_DISP_OFF (0b00000000)
- #define HTK16K33_REG_DIM_1 (0b11100000)
- #define HTK16K33_REG_DIM_2 (0b11100001)
- #define HTK16K33_REG_DIM_3 (0b11100010)
- #define HTK16K33_REG_DIM_4 (0b11100011)
- #define HTK16K33_REG_DIM_5 (0b11100100)
- #define HTK16K33_REG_DIM_6 (0b11100101)
- #define HTK16K33_REG_DIM_7 (0b11100110)
- #define HTK16K33_REG_DIM_8 (0b11100111)
- #define HTK16K33_REG_DIM_9 (0b11101000)
- #define HTK16K33_REG_DIM_10 (0b11101001)
- #define HTK16K33_REG_DIM_11 (0b11101010)
- #define HTK16K33_REG_DIM_12 (0b11101011)
- #define HTK16K33_REG_DIM_13 (0b11101100)
- #define HTK16K33_REG_DIM_14 (0b11101101)
- #define HTK16K33_REG_DIM_15 (0b11101110)
- #define HTK16K33_REG_DIM_16 (0b11101111)

Functions

- int ht16k33_init (uint8_t brightness)
- int ht16k33_clear (void)
- int ht16k33_write_cmd (uint8_t reg)
- int ht16k33_write_num (uint8_t addr, uint8_t value, bool dec)
- int ht16k33_write_hex (uint8_t addr, uint8_t value, bool dec)
- int ht16k33_write_alpha (uint8_t addr, uint8_t value, bool dec)

5.2 ht16k33.h

```
00001 /*
00002  * Licensed to the Apache Software Foundation (ASF) under one
00003  * or more contributor license agreements.  See the NOTICE file
00004  * distributed with this work for additional information
00005  * regarding copyright ownership.  The ASF licenses this file
00006  * to you under the Apache License, Version 2.0 (the
00007  * "License"); you may not use this file except in compliance
00008  * with the License.  You may obtain a copy of the License at
00009  *
00010  *  http://www.apache.org/licenses/LICENSE-2.0
00011  *
00012  * Unless required by applicable law or agreed to in writing,
00013  * software distributed under the License is distributed on an
00014  * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
00015  * KIND, either express or implied.  See the License for the
00016  * specific language governing permissions and limitations
00017  * under the License.
00018  */
00019
00026  #ifndef _MB_HT16K33_H_
00027  #define _MB_HT16K33_H_
00028
00029  #include <stdint.h>
00030  #include "os/mynewt.h"
00031
00041  #define HT16K33_REG_CMD_MODE_ON              (0b00100001)
00042
00044  #define HT16K33_REG_CMD_MODE_STDBY           (0b00100000)
00045
00047  #define HT16K33_REG_DISP_ON_BLINK_NONE       (0b10000001)
00048
00050  #define HT16K33_REG_DISP_ON_BLINK_2HZ        (0b10000011)
00051
00053  #define HT16K33_REG_DISP_ON_BLINK_1HZ        (0b10000101)
00054
00056  #define HT16K33_REG_DISP_ON_BLINK_0_5HZ      (0b10000111)
00057
00059  #define HT16K33_REG_DISP_OFF                 (0b00000000)
00060
00061  /* Sets the display dimming duty cycle (brightness) to 1/16. */
00062  #define HTK16K33_REG_DIM_1                   (0b11100000)
00063
00064  /* Sets the display dimming duty cycle (brightness) to 2/16. */
00065  #define HTK16K33_REG_DIM_2                   (0b11100001)
00066
00067  /* Sets the display dimming duty cycle (brightness) to 3/16. */
00068  #define HTK16K33_REG_DIM_3                   (0b11100010)
00069
00070  /* Sets the display dimming duty cycle (brightness) to 4/16. */
00071  #define HTK16K33_REG_DIM_4                   (0b11100011)
00072
00073  /* Sets the display dimming duty cycle (brightness) to 5/16. */
00074  #define HTK16K33_REG_DIM_5                   (0b11100100)
00075
00076  /* Sets the display dimming duty cycle (brightness) to 6/16. */
00077  #define HTK16K33_REG_DIM_6                   (0b11100101)
00078
00079  /* Sets the display dimming duty cycle (brightness) to 7/16. */
00080  #define HTK16K33_REG_DIM_7                   (0b11100110)
00081
00082  /* Sets the display dimming duty cycle (brightness) to 8/16. */
00083  #define HTK16K33_REG_DIM_8                   (0b11100111)
00084
00085  /* Sets the display dimming duty cycle (brightness) to 9/16. */
00086  #define HTK16K33_REG_DIM_9                   (0b11101000)
00087
00088  /* Sets the display dimming duty cycle (brightness) to 10/16. */
```

```
00089  #define HTK16K33_REG_DIM_10                  (0b11101001)
00090
00091  /* Sets the display dimming duty cycle (brightness) to 11/16. */
00092  #define HTK16K33_REG_DIM_11                  (0b11101010)
00093
00094  /* Sets the display dimming duty cycle (brightness) to 12/16. */
00095  #define HTK16K33_REG_DIM_12                  (0b11101011)
00096
00097  /* Sets the display dimming duty cycle (brightness) to 13/16. */
00098  #define HTK16K33_REG_DIM_13                  (0b11101100)
00099
00100  /* Sets the display dimming duty cycle (brightness) to 14/16. */
00101  #define HTK16K33_REG_DIM_14                  (0b11101101)
00102
00103  /* Sets the display dimming duty cycle (brightness) to 15/16. */
00104  #define HTK16K33_REG_DIM_15                  (0b11101110)
00105
00106  /* Sets the display dimming duty cycle (brightness) to 16/16. */
00107  #define HTK16K33_REG_DIM_16                  (0b11101111)
00108  /* End of DEFINES group */
00110
00111  #ifdef __cplusplus
00112  extern "C" {
00113  #endif
00114
00130 int ht16k33_init(uint8_t brightness);
00131
00132
00138 int ht16k33_clear(void);
00139
00147 int ht16k33_write_cmd(uint8_t reg);
00148
00158 int ht16k33_write_num(uint8_t addr, uint8_t value, bool dec);
00159
00169 int ht16k33_write_hex(uint8_t addr, uint8_t value, bool dec);
00170
00180 int ht16k33_write_alpha(uint8_t addr, uint8_t value, bool dec);
00181
00182 #if MYNEWT_VAL(HT16K33_CLI)
00183
00188 int ht16k33_shell_init(void);
00189 #endif
00190  /* End of FUNC group */
00192
00193 #ifdef __cplusplus
00194 }
00195 #endif
00196
00197 #endif /* _MB_HT16K33_H_ */
00198  /* End of HT16K33 group */
```

5.3 /Users/kevin/Dropbox/microBuilder/Code/nRF52/Mynewt/garden/repos/mb_ht16k33/README.md File Reference

5.4 /Users/kevin/Dropbox/microBuilder/Code/nRF52/Mynewt/garden/repos/mb_ht16k33/README.md

```
00001 # HT16K33 14-Segment Display Driver for Apache Mynewt
00002
00003 The main package documentation for this repository is available at
00004 [microbuilder.io/mb_ht16k33](http://microbuilder.io/mb_ht16k33).
00005
00006 It has been written against Apache Mynewt 1.4.1.
00007
00008 # Folder Structure
00009
00010 - Reasonably complete **documentation** is available in the `docs` folder.
00011
00012 # Hardware
00013
00014 This package has been tested with the following hardware:
00015
00016 - [Adafruit Feather nRF52 Pro with myNewt Bootloader -
       nRF52832](https://www.adafruit.com/product/3574)
00017
00018 - [Adafruit 14-Segment Alphanumeric LED FeatherWing](https://www.adafruit.com/product/3089)
```

```
00019
00020 - [Dual Alphanumeric Display -Yellow 0.54" Digit Height - Pack of
       2](https://www.adafruit.com/product/2154)
00021
00022 # Usage
00023
00024 The following code snippet shows how this package can be used. It assumes the
00025 presence of 4 14-segment displays, numbered 0..3 below:
00026
00027 ```
00028 int rc;
00029
00030 /* Initialise the display at brightness 9/16. */
00031 rc = ht16k33_init(HTK16K33_REG_DIM_9);
00032 if (rc) {
00033     console_printf("HT16K33 init failed: %d\n", rc);
00034 }
00035
00036 /* Display some test data on the display. */
00037 ht16k33_write_num(0, 1, true);      /* 1. */
00038 ht16k33_write_num(1, 2, false);     /* 2  */
00039 ht16k33_write_num(2, 5, false);     /* 5  */
00040 ht16k33_write_alpha(3, 'K', false); /* K  */
00041 ```
```

5.5 /Users/kevin/Dropbox/microBuilder/Code/nRF52/Mynewt/garden/repos/mb_ht16k33/src/ht16k33.c File Reference

```
#include "ht16k33/ht16k33.h"
#include "ht16k33_priv.h"
```

Macros

- #define HT16K33_LOG(lvl_, ...) MODLOG_ ## lvl_(MYNEWT_VAL(HT16K33_LOG_MODULE), __VA_A↩ RGS__)

Functions

- STATS_SECT_END STATS_SECT_DECL (ht16k33_stat_section)
- int ht16k33_i2c_write8 (uint8_t value)
- int ht16k33_i2c_writelen (uint8_t *buffer, uint8_t len)
- int ht16k33_init (uint8_t brightness)
- int ht16k33_clear (void)
- int ht16k33_write_cmd (uint8_t reg)
- int ht16k33_write_num (uint8_t addr, uint8_t value, bool dec)
- int ht16k33_write_hex (uint8_t addr, uint8_t value, bool dec)
- int ht16k33_write_alpha (uint8_t addr, uint8_t value, bool dec)

Variables

- const uint16_t g_ht16k33_tbl_alpha [ ]

5.5.1 Macro Definition Documentation

5.5.1.1 HT16K33_LOG

```
#define HT16K33_LOG(
              lvl_,
              ...  ) MODLOG_ ## lvl_(MYNEWT_VAL(HT16K33_LOG_MODULE), __VA_ARGS__)
```

5.5.2 Function Documentation

5.5.2.1 ht16k33_i2c_write8()

```
int ht16k33_i2c_write8 (
             uint8_t value )
```

Writes a single byte to the HT16K33.

Parameters

value	The value to write to the HT16K33.

Returns

0 o, success, error code on error.

Definition at line 175 of file ht16k33.c.

```
00176 {
00177     int rc;
00178     uint8_t payload[1] = { value };
00179
00180     struct hal_i2c_master_data data_struct = {
00181         .address = MYNEWT_VAL(HT16K33_ITF_ADDR),
00182         .len = 1,
00183         .buffer = payload
00184     };
00185
00186     /* TODO: Add locking interface for I2C access. */
00187
00188     rc = i2cn_master_write(MYNEWT_VAL(HT16K33_ITF_NUM), &data_struct,
00189         OS_TICKS_PER_SEC / 10, 1, MYNEWT_VAL(HT16K33_I2C_RETRIES));
00190     if (rc) {
00191         HT16K33_LOG(ERROR,
00192                     "Failed to write to 0x%02X with value 0x%02lX\n",
00193                     data_struct.address, value);
00194         STATS_INC(g_ht16k33stats, errors);
00195     }
00196
00197     /* TODO: Unlock here. */
00198
00199     return rc;
00200 }
```

5.5.2.2 ht16k33_i2c_writelen()

```
int ht16k33_i2c_writelen (
             uint8_t * buffer,
             uint8_t len )
```

Writes multiple bytes to the HT16K33.

Parameters

buffer	Pointer to the buffer containing the data to write.
len	The number of bytes in the data buffer (max 8!).

Returns

0 o, success, error code on error.

Definition at line 203 of file ht16k33.c.

```
00204 {
00205     int rc;
00206     uint8_t payload[16] = { 0 };
00207
00208     struct hal_i2c_master_data data_struct = {
00209         .address = MYNEWT_VAL(HT16K33_ITF_ADDR),
00210         .len = len,
00211         .buffer = payload
00212     };
00213
00214     if (len > sizeof(payload)) {
00215         rc = OS_EINVAL;
00216         goto err;
00217     }
00218
00219     memcpy(payload, buffer, len);
00220
00221     /* TODO: Add locking interface for I2C access. */
00222
00223     /* Write data */
00224     rc = i2cn_master_write(MYNEWT_VAL(HT16K33_ITF_NUM), &data_struct,
00225         OS_TICKS_PER_SEC / 10, len, MYNEWT_VAL(HT16K33_I2C_RETRIES));
00226     if (rc) {
00227         HT16K33_LOG(ERROR, "I2C access failed at address 0x%02X\n",
00228                     data_struct.address);
00229         STATS_INC(g_ht16k33stats, errors);
00230         goto err;
00231     }
00232
00233 err:
00234     /* TODO: Unlock here. */
00235
00236     return rc;
00237 }
```

5.5.2.3 STATS_SECT_DECL()

```
STATS_SECT_END STATS_SECT_DECL (
             ht16k33_stat_section  )
```

Buffer for the four display characters plus a starting address byte.

Definition at line 34 of file ht16k33.c.

```
00040                                       { 0 };
```

5.5.3 Variable Documentation

5.5.3.1 g_ht16k33_tbl_alpha

```
const uint16_t g_ht16k33_tbl_alpha[]
```

Alpha-numeric lookup table.

Definition at line 43 of file ht16k33.c.

5.6 ht16k33.c

```
00001 /*
00002  * Licensed to the Apache Software Foundation (ASF) under one
00003  * or more contributor license agreements.  See the NOTICE file
00004  * distributed with this work for additional information
00005  * regarding copyright ownership.  The ASF licenses this file
00006  * to you under the Apache License, Version 2.0 (the
00007  * "License"); you may not use this file except in compliance
00008  * with the License.  You may obtain a copy of the License at
00009  *
00010  *  http://www.apache.org/licenses/LICENSE-2.0
00011  *
00012  * Unless required by applicable law or agreed to in writing,
00013  * software distributed under the License is distributed on an
00014  * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
00015  * KIND, either express or implied.  See the License for the
00016  * specific language governing permissions and limitations
00017  * under the License.
00018  */
00019
00020 #include "ht16k33/ht16k33.h"
00021 #include "ht16k33_priv.h"
00022
00023 /* Define the stats section and records */
00024 STATS_SECT_START(ht16k33_stat_section)
00025     STATS_SECT_ENTRY(errors)
00026 STATS_SECT_END
00027
00028 /* Define stat names for querying */
00029 STATS_NAME_START(ht16k33_stat_section)
00030     STATS_NAME(ht16k33_stat_section, errors)
00031 STATS_NAME_END(ht16k33_stat_section)
00032
00033 /* Global variable used to hold stats data */
00034 STATS_SECT_DECL(ht16k33_stat_section) g_ht16k33stats;
00035
00036 #define HT16K33_LOG(lvl_, ...) \
00037     MODLOG_ ## lvl_(MYNEWT_VAL(HT16K33_LOG_MODULE), __VA_ARGS__)
00038
00040 static uint8_t g_ht16k33_buffer_16[9] = { 0 };
00041
00043 const uint16_t g_ht16k33_tbl_alpha[] = {
00044     0b0000000000000001,
00045     0b0000000000000010,
00046     0b0000000000000100,
00047     0b0000000000001000,
00048     0b0000000000010000,
00049     0b0000000000100000,
00050     0b0000000001000000,
00051     0b0000000010000000,
00052     0b0000000100000000,
00053     0b0000001000000000,
00054     0b0000010000000000,
00055     0b0000100000000000,
00056     0b0001000000000000,
00057     0b0010000000000000,
00058     0b0100000000000000,
00059     0b1000000000000000,
00060     0b0000000000000000,
00061     0b0000000000000000,
00062     0b0000000000000000,
00063     0b0000000000000000,
00064     0b0000000000000000,
00065     0b0000000000000000,
00066     0b0000000000000000,
00067     0b0000000000000000,
00068     0b0001001011001001,
00069     0b0001010111000000,
00070     0b0001001011111001,
```

```
00071     0b0000000011100011,
00072     0b0000010100110000,
00073     0b0001001011001000,
00074     0b0011101000000000,
00075     0b0001011100000000,
00076     0b0000000000000000, //
00077     0b0000000000000110, // !
00078     0b0000001000100000, // "
00079     0b0001001011001110, // #
00080     0b0001001011101101, // $
00081     0b0000110000100100, // %
00082     0b0010001101011101, // &
00083     0b0000010000000000, // '
00084     0b0010010000000000, // (
00085     0b0000100100000000, // )
00086     0b0011111111000000, // *
00087     0b0001001011000000, // +
00088     0b0000100000000000, // ,
00089     0b0000000011000000, // -
00090     0b0000000000000000, // .
00091     0b0000110000000000, // /
00092     0b0000110000111111, // 0
00093     0b0000000000000110, // 1
00094     0b0000000011011011, // 2
00095     0b0000000010001111, // 3
00096     0b0000000011100110, // 4
00097     0b0010000001101001, // 5
00098     0b0000000011111101, // 6
00099     0b0000000000000111, // 7
00100     0b0000000011111111, // 8
00101     0b0000000011101111, // 9
00102     0b0001001000000000, // :
00103     0b0000101000000000, // ;
00104     0b0010010000000000, // <
00105     0b0000000011001000, // =
00106     0b0000100100000000, // >
00107     0b0001000010000011, // ?
00108     0b0000001010111011, // @
00109     0b0000000011110111, // A
00110     0b0001001010001111, // B
00111     0b0000000000111001, // C
00112     0b0001001000001111, // D
00113     0b0000000011111001, // E
00114     0b0000000001110001, // F
00115     0b0000000010111101, // G
00116     0b0000000011110110, // H
00117     0b0001001000000000, // I
00118     0b0000000000011110, // J
00119     0b0010010001110000, // K
00120     0b0000000000111000, // L
00121     0b0000010100110110, // M
00122     0b0010000100110110, // N
00123     0b0000000000111111, // O
00124     0b0000000011110011, // P
00125     0b0010000000111111, // Q
00126     0b0010000011110011, // R
00127     0b0000000011101101, // S
00128     0b0001001000000001, // T
00129     0b0000000000111110, // U
00130     0b0000110000110000, // V
00131     0b0010100000110110, // W
00132     0b0010110100000000, // X
00133     0b0001010100000000, // Y
00134     0b0000110000001001, // Z
00135     0b0000000000111001, // [
00136     0b0010000100000000, //
00137     0b0000000000001111, // ]
00138     0b0000110000000011, // ^
00139     0b0000000000001000, // _
00140     0b0000000100000000, // `
00141     0b0001000001011000, // a
00142     0b0010000001111000, // b
00143     0b0000000011011000, // c
00144     0b0000100010001110, // d
00145     0b0000100001011000, // e
00146     0b0000000001110001, // f
00147     0b0000010010001110, // g
00148     0b0001000001110000, // h
00149     0b0001000000000000, // i
00150     0b0000000000001110, // j
00151     0b0011011000000000, // k
00152     0b0000000000110000, // l
00153     0b0001000011010100, // m
00154     0b0001000001010000, // n
00155     0b0000000011011100, // o
00156     0b0000000101110000, // p
00157     0b0000010010000110, // q
```

```
00158     0b0000000001010000, // r
00159     0b0010000010001000, // s
00160     0b0000000001111000, // t
00161     0b0000000000011100, // u
00162     0b0010000000000100, // v
00163     0b0010100000010100, // w
00164     0b0010100011000000, // x
00165     0b0010000000001100, // y
00166     0b0000100001001000, // z
00167     0b0000100101001001, // {
00168     0b0001001000000000, // |
00169     0b0010010010001001, // }
00170     0b0000010100100000, // ~
00171     0b0011111111111111,
00172 };
00173
00174 int
00175 ht16k33_i2c_write8(uint8_t value)
00176 {
00177     int rc;
00178     uint8_t payload[1] = { value };
00179
00180     struct hal_i2c_master_data data_struct = {
00181         .address = MYNEWT_VAL(HT16K33_ITF_ADDR),
00182         .len = 1,
00183         .buffer = payload
00184     };
00185
00186     /* TODO: Add locking interface for I2C access. */
00187
00188     rc = i2cn_master_write(MYNEWT_VAL(HT16K33_ITF_NUM), &data_struct,
00189         OS_TICKS_PER_SEC / 10, 1, MYNEWT_VAL(HT16K33_I2C_RETRIES));
00190     if (rc) {
00191         HT16K33_LOG(ERROR,
00192                     "Failed to write to 0x%02X with value 0x%02lX\n",
00193                     data_struct.address, value);
00194         STATS_INC(g_ht16k33stats, errors);
00195     }
00196
00197     /* TODO: Unlock here. */
00198
00199     return rc;
00200 }
00201
00202 int
00203 ht16k33_i2c_writelen(uint8_t *buffer, uint8_t len)
00204 {
00205     int rc;
00206     uint8_t payload[16] = { 0 };
00207
00208     struct hal_i2c_master_data data_struct = {
00209         .address = MYNEWT_VAL(HT16K33_ITF_ADDR),
00210         .len = len,
00211         .buffer = payload
00212     };
00213
00214     if (len > sizeof(payload)) {
00215         rc = OS_EINVAL;
00216         goto err;
00217     }
00218
00219     memcpy(payload, buffer, len);
00220
00221     /* TODO: Add locking interface for I2C access. */
00222
00223     /* Write data */
00224     rc = i2cn_master_write(MYNEWT_VAL(HT16K33_ITF_NUM), &data_struct,
00225         OS_TICKS_PER_SEC / 10, len, MYNEWT_VAL(HT16K33_I2C_RETRIES));
00226     if (rc) {
00227         HT16K33_LOG(ERROR, "I2C access failed at address 0x%02X\n",
00228                     data_struct.address);
00229         STATS_INC(g_ht16k33stats, errors);
00230         goto err;
00231     }
00232
00233 err:
00234     /* TODO: Unlock here. */
00235
00236     return rc;
00237 }
00238
00239 int
00240 ht16k33_init(uint8_t brightness)
00241 {
00242     int rc;
00243
00244     /* Insert a short delay to allow the IC to start up. */
```

```
00245     os_time_delay((1 * OS_TICKS_PER_SEC)/1000);
00246
00247     /* Initialise the stats entry. */
00248     rc = stats_init(
00249         STATS_HDR(g_ht16k33stats),
00250         STATS_SIZE_INIT_PARMS(g_ht16k33stats, STATS_SIZE_32),
00251         STATS_NAME_INIT_PARMS(ht16k33_stat_section));
00252     if (rc != 0) {
00253         goto err;
00254     }
00255
00256     /* Register the entry with the stats registry. */
00257     rc = stats_register("ht16k33", STATS_HDR(g_ht16k33stats));
00258     if (rc != 0) {
00259         goto err;
00260     }
00261
00262     /* Enable the controller. */
00263     rc = ht16k33_write_cmd(HT16K33_REG_CMD_MODE_ON);
00264     if (rc != 0) {
00265         goto err;
00266     }
00267
00268     /* Clear the display. */
00269     rc = ht16k33_clear();
00270     if (rc != 0) {
00271         goto err;
00272     }
00273
00274     /* Turn the display on. */
00275     rc = ht16k33_write_cmd(HT16K33_REG_DISP_ON_BLINK_NONE);
00276     if (rc != 0) {
00277         goto err;
00278     }
00279
00280     /* Set the display brightness */
00281     rc = ht16k33_write_cmd(brightness);
00282     if (rc != 0) {
00283         goto err;
00284     }
00285
00286     return (0);
00287 err:
00288     return (rc);
00289 }
00290
00291 int
00292 ht16k33_clear(void)
00293 {
00294     int rc;
00295
00296     /* Clear the buffer. */
00297     memset(g_ht16k33_buffer_16, 0, sizeof(g_ht16k33_buffer_16));
00298
00299     rc = ht16k33_i2c_writelen(g_ht16k33_buffer_16,
00300         sizeof(g_ht16k33_buffer_16));
00301     if (rc != 0) {
00302         goto err;
00303     }
00304
00305     return 0;
00306 err:
00307     return rc;
00308 }
00309
00310 int
00311 ht16k33_write_cmd(uint8_t reg)
00312 {
00313     int rc;
00314
00315     rc = ht16k33_i2c_write8(reg);
00316     if (rc != 0) {
00317         goto err;
00318     }
00319
00320     return 0;
00321 err:
00322     return rc;
00323 }
00324
00325 int
00326 ht16k33_write_num(uint8_t addr, uint8_t value, bool dec)
00327 {
00328     int rc;
00329
00330     /* Make sure we stay in range. */
00331     if (value > 9) {
```

```
00332         rc = OS_EINVAL;
00333         goto err;
00334     }
00335 
00336     /* Set the address and hex data. */
00337     g_ht16k33_buffer_16[addr*2+1] = (g_ht16k33_tbl_alpha[value + 48] & 0xFF);
00338     g_ht16k33_buffer_16[addr*2+2] = (g_ht16k33_tbl_alpha[value + 48]>>8) & 0xFF;
00339 
00340     /* Add the decimal point to the output if requested. */
00341     if (dec) {
00342         g_ht16k33_buffer_16[addr*2+2] |= 0x40;
00343     }
00344 
00345     rc = ht16k33_i2c_writelen(g_ht16k33_buffer_16, sizeof(g_ht16k33_buffer_16));
00346     if (rc != 0) {
00347         goto err;
00348     }
00349 
00350     return 0;
00351 err:
00352     return rc;
00353 }
00354 
00355 int
00356 ht16k33_write_hex(uint8_t addr, uint8_t value, bool dec)
00357 {
00358     int rc;
00359 
00360     /* Make sure we stay in range. */
00361     if (value > 0xF) {
00362         rc = OS_EINVAL;
00363         goto err;
00364     }
00365 
00366     if (value < 10) {
00367         /* Handle decimal values 0..9 */
00368         return ht16k33_write_num(addr, value, dec);
00369     } else {
00370         /* Display HEX value as ASCII */
00371         return ht16k33_write_alpha(addr, value+55, dec);
00372     }
00373 
00374 err:
00375     return rc;
00376 }
00377 
00378 int
00379 ht16k33_write_alpha(uint8_t addr, uint8_t value, bool dec)
00380 {
00381     int rc;
00382 
00383     /* Make sure we stay in range. */
00384     if (value >= 128) {
00385         rc = OS_EINVAL;
00386         goto err;
00387     }
00388 
00389     /* Set the address and hex data. */
00390     g_ht16k33_buffer_16[addr*2+1] = (g_ht16k33_tbl_alpha[value] & 0xFF);
00391     g_ht16k33_buffer_16[addr*2+2] = (g_ht16k33_tbl_alpha[value] >> 8) & 0xFF;
00392 
00393     /* Add the decimal point to the output if requested. */
00394     if (dec) {
00395         g_ht16k33_buffer_16[addr*2+2] |= 0x40;
00396     }
00397 
00398     rc = ht16k33_i2c_writelen(g_ht16k33_buffer_16, sizeof(g_ht16k33_buffer_16));
00399     if (rc != 0) {
00400         goto err;
00401     }
00402 
00403     return 0;
00404 err:
00405     return rc;
00406 }
```

5.7 /Users/kevin/Dropbox/microBuilder/Code/nRF52/Mynewt/garden/repos/mb_ht16k33/src/ht16k33_priv.h File Reference

```
#include "os/mynewt.h"
#include "hal/hal_i2c.h"
```

```
#include "i2cn/i2cn.h"
#include "modlog/modlog.h"
#include "stats/stats.h"
#include "syscfg/syscfg.h"
#include "ht16k33/ht16k33.h"
```

Functions

- int ht16k33_i2c_write8 (uint8_t value)
- int ht16k33_i2c_writelen (uint8_t *buffer, uint8_t len)

5.7.1 Function Documentation

5.7.1.1 ht16k33_i2c_write8()

```
int ht16k33_i2c_write8 (
            uint8_t value )
```

Writes a single byte to the HT16K33.

Parameters

value	The value to write to the HT16K33.

Returns

0 o, success, error code on error.

Definition at line 175 of file ht16k33.c.

```
00176 {
00177     int rc;
00178     uint8_t payload[1] = { value };
00179
00180     struct hal_i2c_master_data data_struct = {
00181         .address = MYNEWT_VAL(HT16K33_ITF_ADDR),
00182         .len = 1,
00183         .buffer = payload
00184     };
00185
00186     /* TODO: Add locking interface for I2C access. */
00187
00188     rc = i2cn_master_write(MYNEWT_VAL(HT16K33_ITF_NUM), &data_struct,
00189         OS_TICKS_PER_SEC / 10, 1, MYNEWT_VAL(HT16K33_I2C_RETRIES));
00190     if (rc) {
00191         HT16K33_LOG(ERROR,
00192                     "Failed to write to 0x%02X with value 0x%02lX\n",
00193                     data_struct.address, value);
00194         STATS_INC(g_ht16k33stats, errors);
00195     }
00196
00197     /* TODO: Unlock here. */
00198
00199     return rc;
00200 }
```

5.7.1.2 ht16k33_i2c_writelen()

```
int ht16k33_i2c_writelen (
            uint8_t * buffer,
            uint8_t len )
```

Writes multiple bytes to the HT16K33.

Parameters

buffer	Pointer to the buffer containing the data to write.
len	The number of bytes in the data buffer (max 8!).

Returns

0 o, success, error code on error.

Definition at line 203 of file ht16k33.c.

```
00204 {
00205     int rc;
00206     uint8_t payload[16] = { 0 };
00207
00208     struct hal_i2c_master_data data_struct = {
00209         .address = MYNEWT_VAL(HT16K33_ITF_ADDR),
00210         .len = len,
00211         .buffer = payload
00212     };
00213
00214     if (len > sizeof(payload)) {
00215         rc = OS_EINVAL;
00216         goto err;
00217     }
00218
00219     memcpy(payload, buffer, len);
00220
00221     /* TODO: Add locking interface for I2C access. */
00222
00223     /* Write data */
00224     rc = i2cn_master_write(MYNEWT_VAL(HT16K33_ITF_NUM), &data_struct,
00225         OS_TICKS_PER_SEC / 10, len, MYNEWT_VAL(HT16K33_I2C_RETRIES));
00226     if (rc) {
00227         HT16K33_LOG(ERROR, "I2C access failed at address 0x%02X\n",
00228                     data_struct.address);
00229         STATS_INC(g_ht16k33stats, errors);
00230         goto err;
00231     }
00232
00233 err:
00234     /* TODO: Unlock here. */
00235
00236     return rc;
00237 }
```

5.8 ht16k33_priv.h

```
00001 /*
00002  * Licensed to the Apache Software Foundation (ASF) under one
00003  * or more contributor license agreements.  See the NOTICE file
00004  * distributed with this work for additional information
00005  * regarding copyright ownership.  The ASF licenses this file
00006  * to you under the Apache License, Version 2.0 (the
00007  * "License"); you may not use this file except in compliance
00008  * with the License.  You may obtain a copy of the License at
00009  *
00010  *  http://www.apache.org/licenses/LICENSE-2.0
00011  *
```

```
00012  * Unless required by applicable law or agreed to in writing,
00013  * software distributed under the License is distributed on an
00014  * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
00015  * KIND, either express or implied.  See the License for the
00016  * specific language governing permissions and limitations
00017  * under the License.
00018  */
00019
00020 #ifndef __HT16K33_PRIV_H__
00021 #define __HT16K33_PRIV_H__
00022
00023 #include "os/mynewt.h"
00024 #include "hal/hal_i2c.h"
00025 #include "i2cn/i2cn.h"
00026 #include "modlog/modlog.h"
00027 #include "stats/stats.h"
00028 #include "syscfg/syscfg.h"
00029 #include "ht16k33/ht16k33.h"
00030
00031 #ifdef __cplusplus
00032 extern "C" {
00033 #endif
00034
00042 int ht16k33_i2c_write8(uint8_t value);
00043
00052 int ht16k33_i2c_writelen(uint8_t *buffer, uint8_t len);
00053
00054 #ifdef __cplusplus
00055 }
00056 #endif
00057
00058 #endif /* __HT16K33_PRIV_H__ */
```

5.9 /Users/kevin/Dropbox/microBuilder/Code/nRF52/Mynewt/garden/repos/mb_ht16k33/src/ht16k33_shell.c File Reference

Shell commands for the 'clr' command..

```
#include "os/mynewt.h"
#include <stdlib.h>
#include <string.h>
#include <errno.h>
#include "console/console.h"
#include "ht16k33/ht16k33.h"
#include "ht16k33_priv.h"
```

5.9.1 Detailed Description

Shell commands for the 'clr' command..

Definition in file ht16k33_shell.c.

5.10 ht16k33_shell.c

```
00001 /*
00002  * Licensed to the Apache Software Foundation (ASF) under one
00003  * or more contributor license agreements.  See the NOTICE file
00004  * distributed with this work for additional information
00005  * regarding copyright ownership.  The ASF licenses this file
00006  * to you under the Apache License, Version 2.0 (the
00007  * "License"); you may not use this file except in compliance
00008  * with the License.  You may obtain a copy of the License at
00009  *
```

```
00010  *  http://www.apache.org/licenses/LICENSE-2.0
00011  *
00012  * Unless required by applicable law or agreed to in writing,
00013  * software distributed under the License is distributed on an
00014  * "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY
00015  * KIND, either express or implied.  See the License for the
00016  * specific language governing permissions and limitations
00017  * under the License.
00018  */
00019
00025 #include "os/mynewt.h"
00026
00027 #include <stdlib.h>
00028 #include <string.h>
00029 #include <errno.h>
00030 #include "console/console.h"
00031 #include "ht16k33/ht16k33.h"
00032 #include "ht16k33_priv.h"
00033
00034 #if MYNEWT_VAL(HT16K33_CLI)
00035
00036 /* These need to be masked out for unit tests to run */
00037 #include "shell/shell.h"
00038 #include "parse/parse.h"
00039
00040 static int ht16k33_shell_cmd(int argc, char **argv);
00041
00042 static struct shell_cmd ht16k33_shell_cmd_struct = {
00043     .sc_cmd = "ht16k33",
00044     .sc_cmd_func = ht16k33_shell_cmd
00045 };
00046
00047 #if 0
00048 static int
00049 ht16k33_shell_err_too_many_args(char *cmd_name)
00050 {
00051     console_printf("Error: too many arguments for command \"%s\"\n",
00052                    cmd_name);
00053     return EINVAL;
00054 }
00055 #endif
00056
00057 #if 0
00058 static int
00059 ht16k33_shell_err_missing_arg(char *arg_name)
00060 {
00061     console_printf("Error: missing arg [%s]\n",
00062                    arg_name);
00063     return EINVAL;
00064 }
00065 #endif
00066
00067 static int
00068 ht16k33_shell_err_unknown_arg(char *cmd_name)
00069 {
00070     console_printf("Error: unknown argument \"%s\"\n",
00071                    cmd_name);
00072     return EINVAL;
00073 }
00074
00075 static int
00076 ht16k33_shell_help(void)
00077 {
00078     console_printf("%s cmd [params...]\n", ht16k33_shell_cmd_struct.sc_cmd);
00079     console_printf("cmd:\n");
00080     console_printf("  todo    todo description\n");
00081
00082     return 0;
00083 }
00084
00085 static int
00086 ht16k33_shell_todo_help(void)
00087 {
00088     console_printf("%s todo cmd [params...]\n", ht16k33_shell_cmd_struct.sc_cmd);
00089     console_printf("cmd:\n");
00090     console_printf("  todo <one> <two>      todo description\n");
00091
00092     return 0;
00093 }
00094
00095 static int
00096 ht16k33_shell_todo(int argc, char **argv)
00097 {
00098     if (argc == 2) {
00099         return ht16k33_shell_todo_help();
00100     }
00101
```

```
00102 #if 0
00103     /* cct2XYZ */
00104     if (argc > 2 && strcmp(argv[2], "cct2XYZ") == 0) {
00105         return clr_shell_conv_cct_XYZ(argc, argv);
00106     }
00107 #endif
00108
00109     /* XYZ2cct */
00110     return ht16k33_shell_err_unknown_arg(argv[2]);
00111 }
00112
00113 static int
00114 ht16k33_shell_cmd(int argc, char **argv)
00115 {
00116     if (argc == 1) {
00117         return ht16k33_shell_help();
00118     }
00119
00120     /* 'todo' sub-command */
00121     if (argc > 1 && strcmp(argv[1], "todo") == 0) {
00122         return ht16k33_shell_todo(argc, argv);
00123     }
00124
00125     return ht16k33_shell_err_unknown_arg(argv[1]);
00126 }
00127
00128 int
00129 ht16k33_shell_init(void)
00130 {
00131     int rc;
00132
00133     rc = shell_cmd_register(&ht16k33_shell_cmd_struct);
00134     SYSINIT_PANIC_ASSERT(rc == 0);
00135
00136     return rc;
00137 }
00138
00139 #endif /* HT16K33_CLI */
```

Index